Nonparametric Regression and the Bootstrap

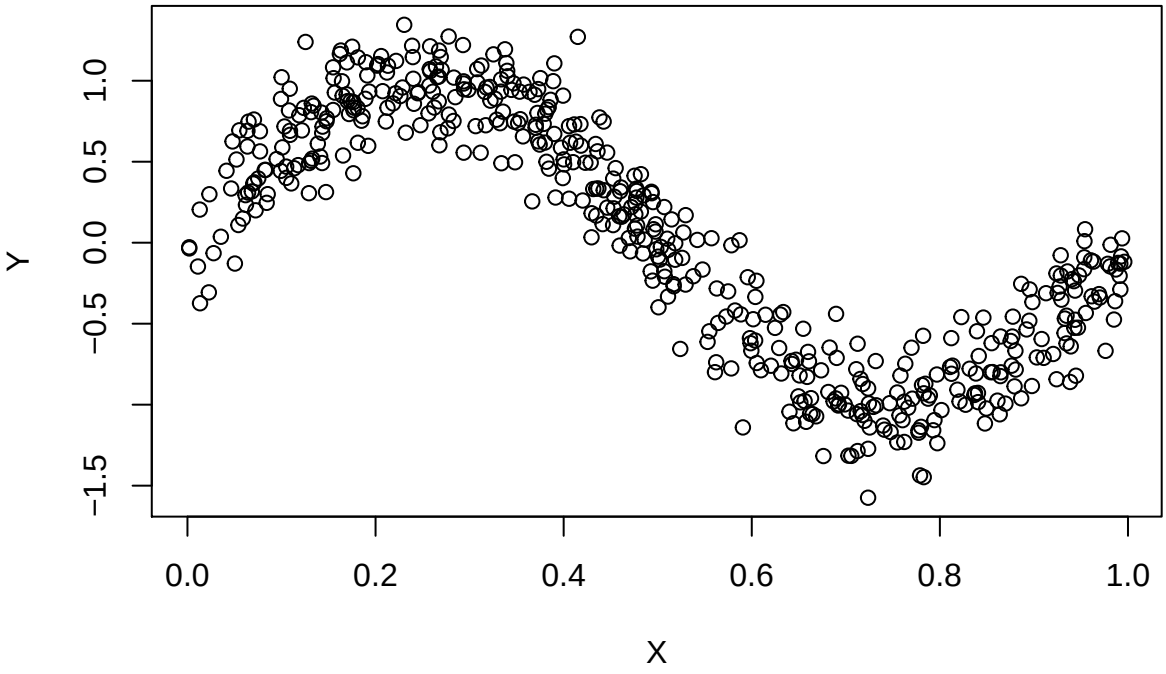
Yen-Chi Chen

December 5, 2016

Nonparametric regression

We first generate a dataset using the following code:

```
set.seed(1)
X <- runif(500)
Y <- sin(X*2*pi)+rnorm(500,sd=0.2)
plot(X,Y)
```



This is a *sine* shape regression dataset.

Simple linear regression

We first fit a linear regression to this dataset:

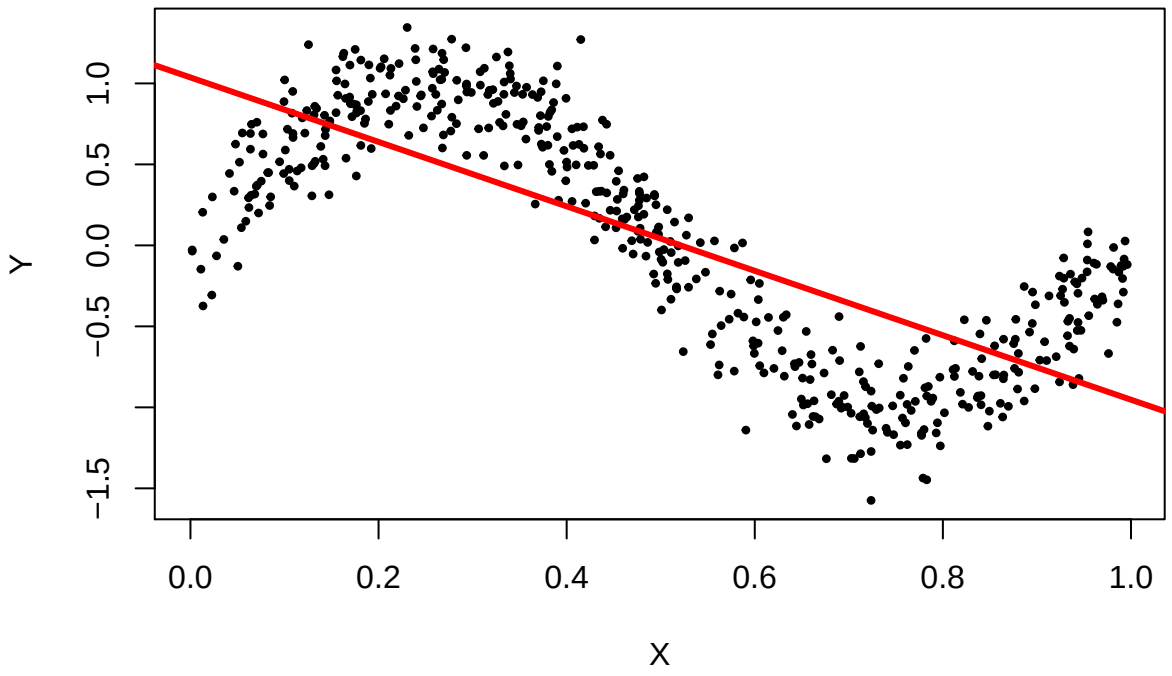
```
fit <- lm(Y~X)
summary(fit)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.38395 -0.35908  0.04589  0.38541  1.05982
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.03681    0.04306   24.08   <2e-16 ***
## X           -1.98962    0.07545  -26.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4775 on 498 degrees of freedom
## Multiple R-squared:  0.5827, Adjusted R-squared:  0.5819
## F-statistic: 695.4 on 1 and 498 DF,  p-value: < 2.2e-16
```

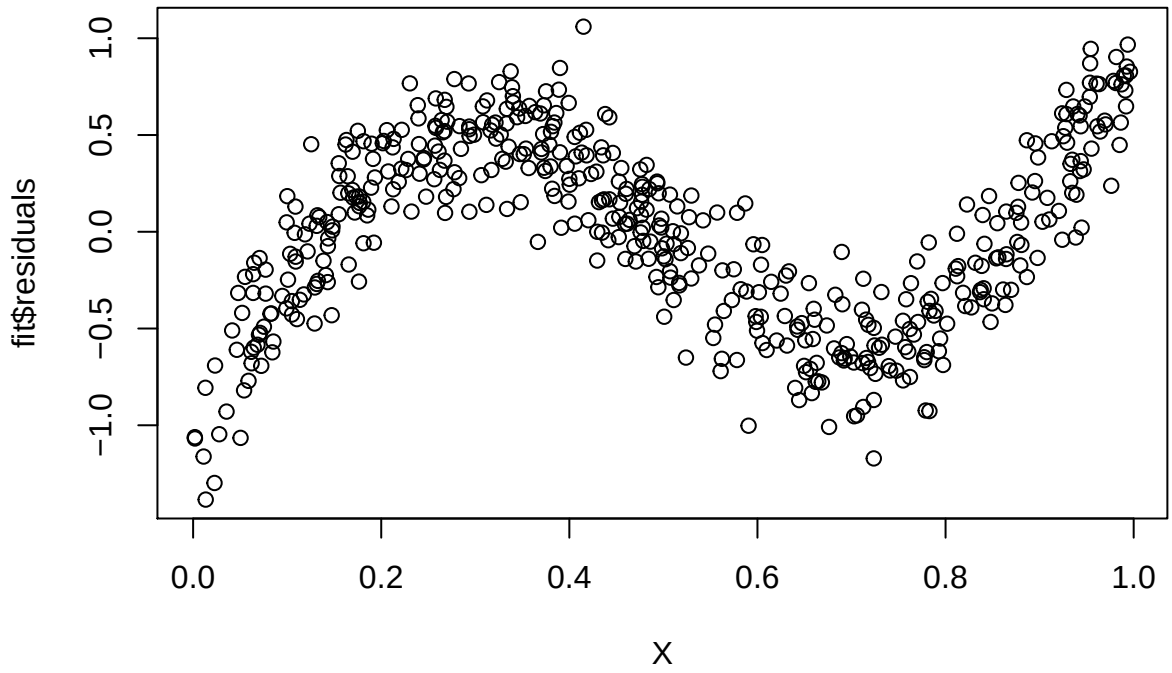
We observe a negative slope and here is the scatter plot with the regression line:

```
plot(X,Y, pch=20, cex=0.7, col="black")
abline(fit, lwd=3, col="red")
```



After fitted a regression, we need to examine the residual plot to see if there is any patetrn left in the fit:

```
plot(X, fit$residuals)
```

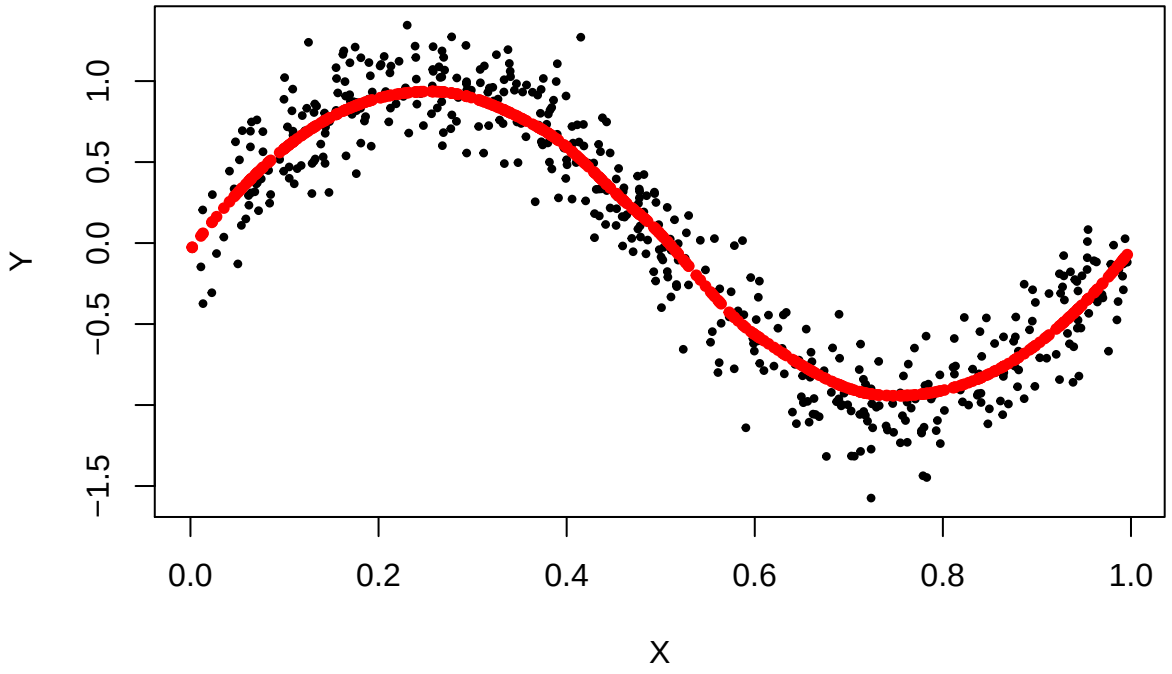


We see a clear pattern in the residual plot–this suggests that there is a nonlinear relationship between `X` and `Y`.

Nonparametric regression: local polynomial regression

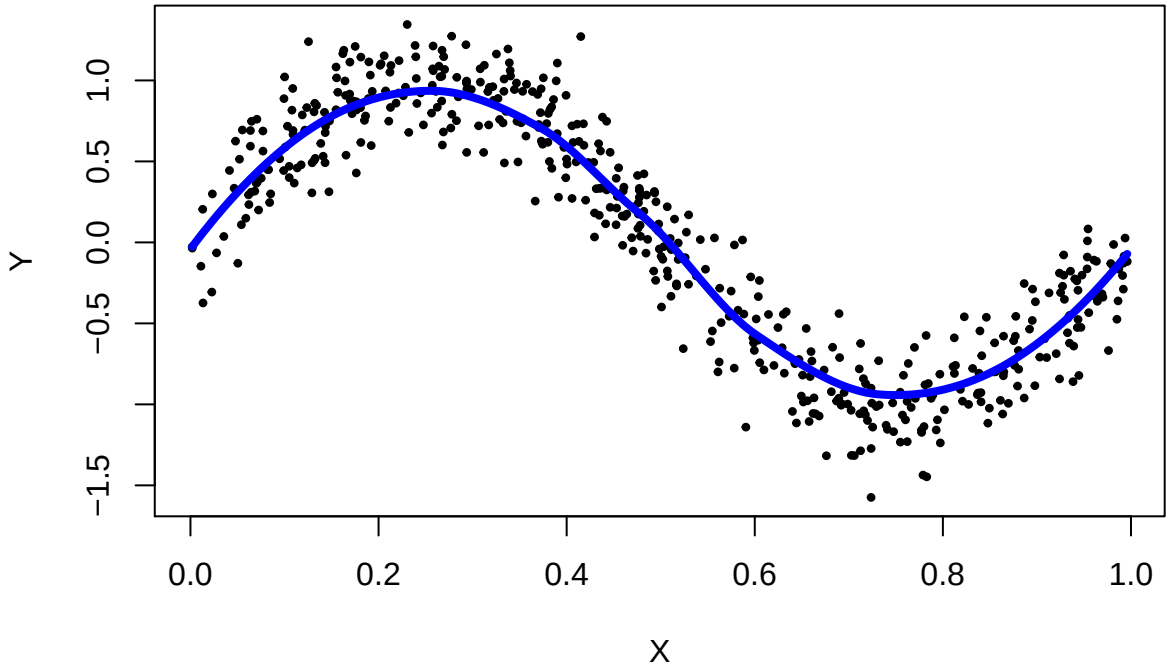
To fit the nonlinear structure, we will use the **nonparametric regression**. Here we apply a method called *local polynomial regression*. In *R*, you can use the function `loess()` to fit the localy polynomial regression.

```
fit2 <- loess(Y~X)
plot(X,Y, pch=20, cex=0.7, col="black")
points(X,fit2$fitted, pch=20, col="red")
```



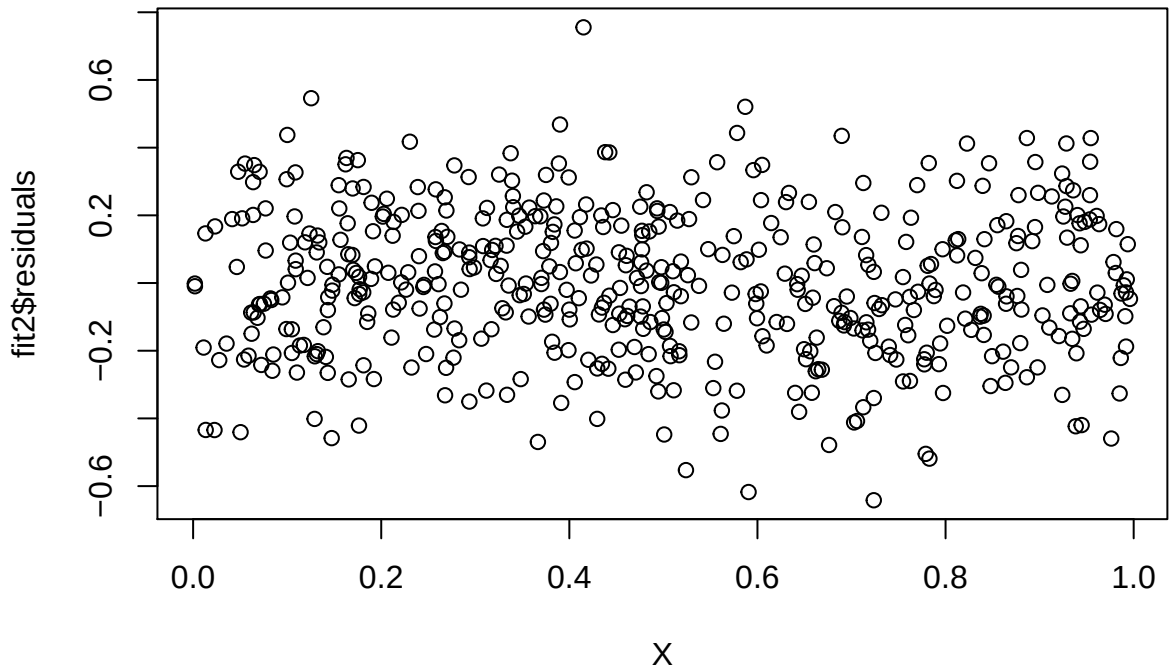
In the above plot, the red dots represent the fitted value at each observed covariate `X`. We can also attach a regression curve to it using the function `line()` but we need to order the data points first.

```
w = order(X)
plot(X,Y, pch=20, cex=0.7, col="black")
lines(X[w],fit2$fitted[w], lwd=4, col="blue")
```



It looks very nice! And here is the residual plot:

```
plot(X, fit2$residuals)
```

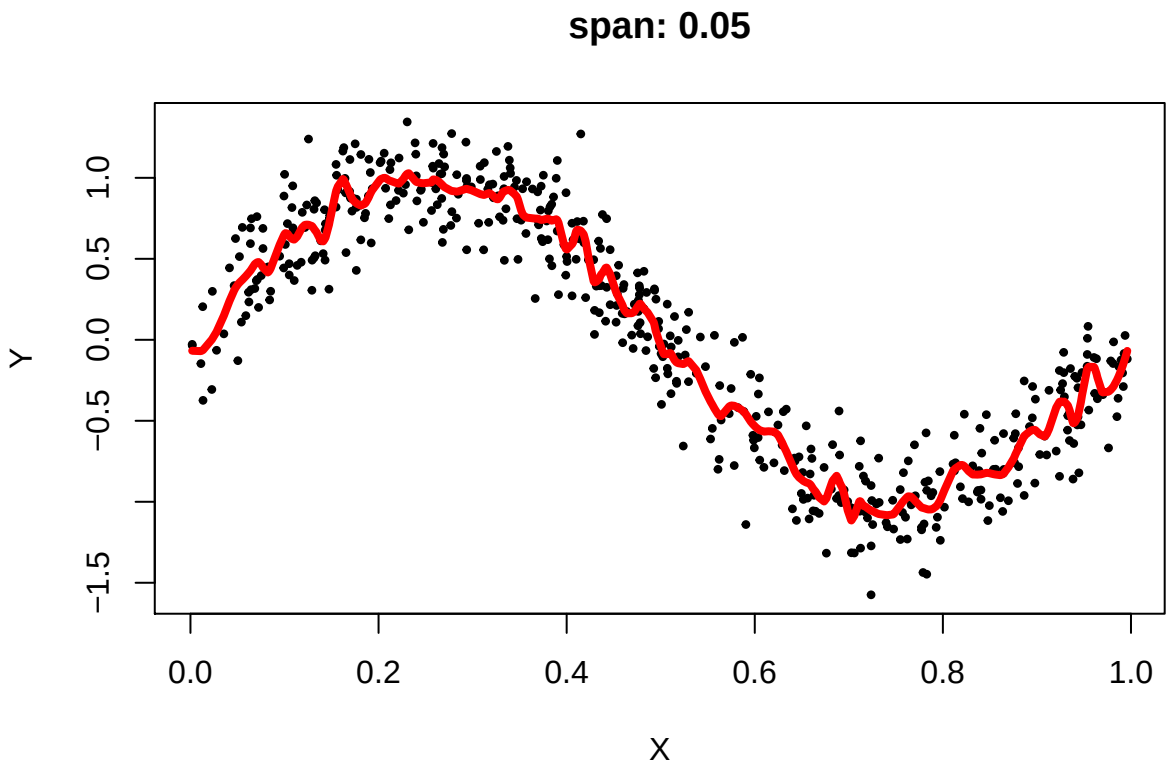


Now we do not observe any pattern inside the residuals. So the fitted curve from nonparametric regression captures most signals in this dataset.

The argument span: smoothing bandwidth

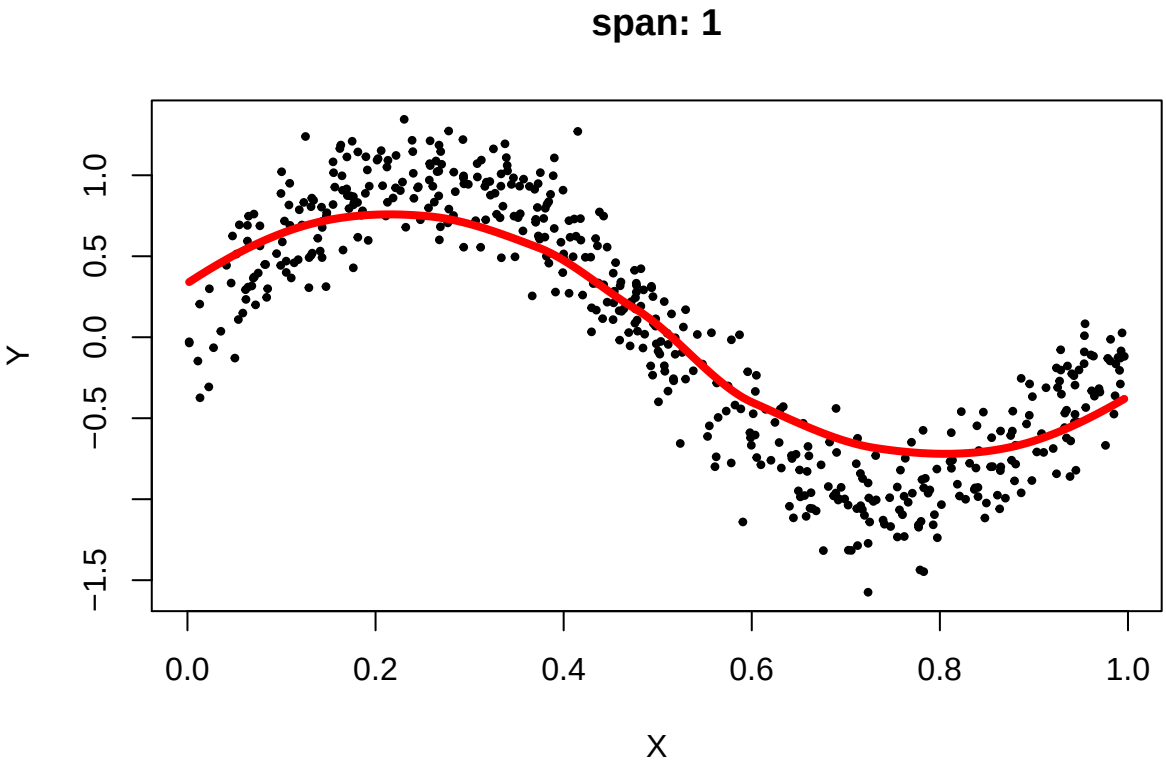
The local polynomial regression is to use a local smoothing window to fit the data. The argument `span` controls the amount of smoothing, which plays a similar role as the smoothing bandwidth in the kernel density estimator (KDE). Here is an example of using a very small smoothing window (undersmoothing):

```
fit2 <- loess(Y~X, span=0.05)
plot(X,Y, pch=20, cex=0.7, col="black",
     main="span: 0.05")
lines(X[w],fit2$fitted[w], lwd=4, col="red")
```



Similar to the KDE, when we undersmoothg the data, we observe lots of wiggling patterns. On the other hand, if we smooth the data too much, we would enter the regime of oversmoothing:

```
fit2 <- loess(Y~X, span=1)
plot(X,Y, pch=20, cex=0.7, col="black",
     main="span: 1")
lines(X[w],fit2$fitted[w], lwd=4, col="red")
```



The oversmoothing leads to a biased structure, the same as the KDE.

- A standard way to choose the smoothing bandwidth is via a method called cross-validation. The basic idea is to split the data into two parts, we use the first part of the data to obtain the fitted curve and estimate the model's errors using the other part of the data.
- Other nonparametric regression methods can be found in wikipedia.

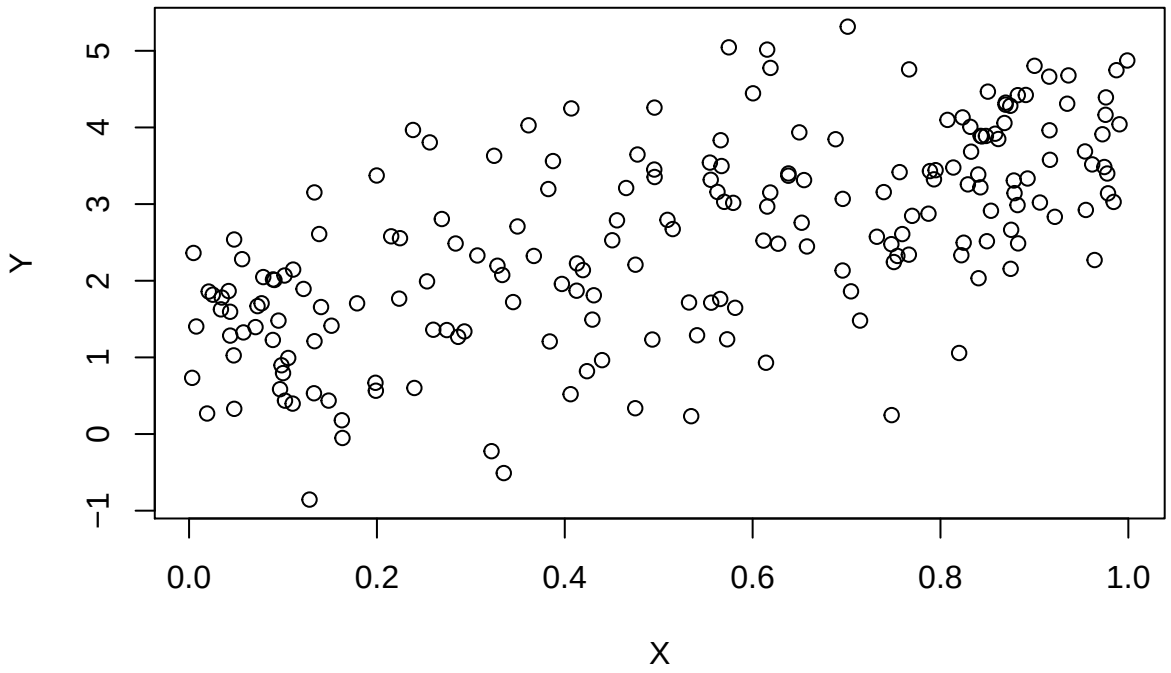
Bootstrap

The other topic for today is a classical statistical method: the bootstrap(https://en.wikipedia.org/wiki/Bootstrapping_(statistics). The bootstrap is a method to approximate the *uncertainty* of our estimator. Before we go to the details of the bootstrap, we first illustrate the uncertainty of an estimator by a simple example.

Uncertainty of an estimator

Because our estimator is computed from the data, it is a random quantity where the randomness is from the sampling procedure that generates the data. Here is an example about the uncertainty of an estimator in the linear regression.

```
X <- runif(200)
Y <- 3*X + 1 + rnorm(200)
plot(X,Y)
```



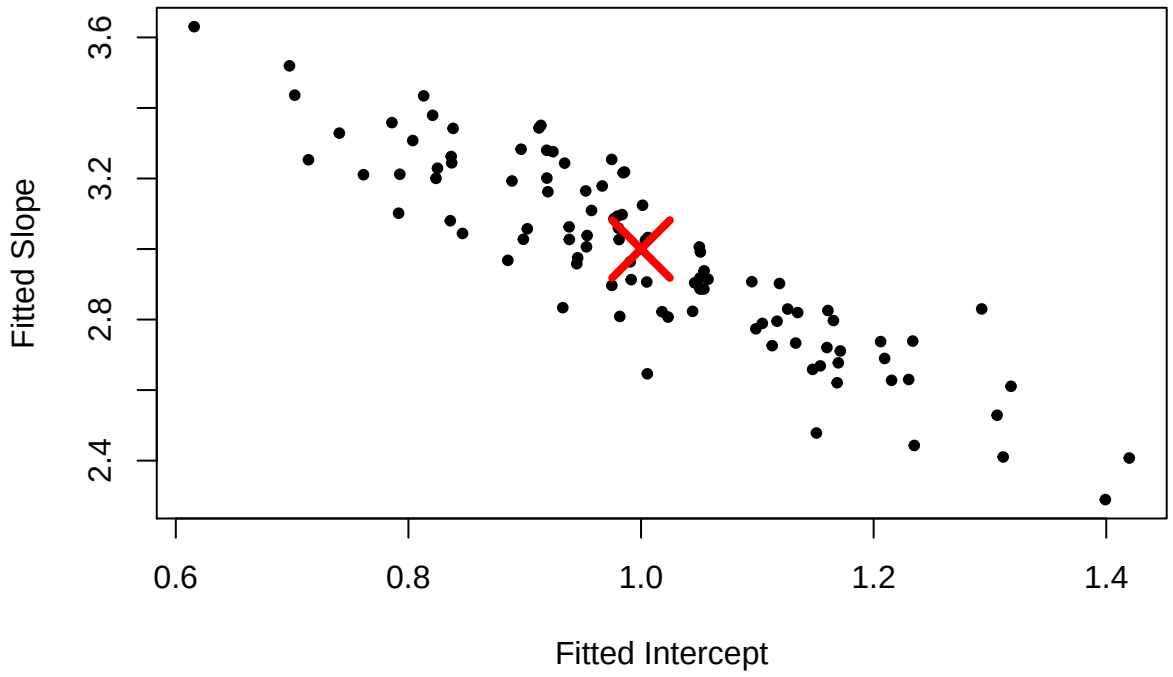

```
fit <- lm(Y~X)
summary(fit)$coefficient
```

```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 1.190728  0.1373085  8.671919 1.533084e-15
## X           2.610872  0.2263429 11.535030 7.035711e-24
```

In the above example, we generate a data with $Y = 3X + 1 + \epsilon$, where the noise ϵ follows a standard normal distribution. The covariate X is from a uniform distribution on the interval $[0, 1]$ and the sample size is 200. We know that the actual slope is 3 and the actual intercept is 1 but the fitted slope and intercept are not the same as the actual slope due to the sampling randomness.

To see how the randomness of the fitted slope and intercept, we repeat the above procedure 100 times and show the fitted values:

```
fitted_table <- matrix(NA, nrow=100, ncol=2)
for(j in 1:100){
  X_new <- runif(200)
  Y_new <- 3*X_new + 1 + rnorm(200)
  fit_new <- lm(Y_new~X_new)
  fitted_table[j,] <- fit_new$coefficients
}
plot(fitted_table, xlab="Fitted Intercept",
     ylab="Fitted Slope", pch=20)
points(x=1, y=3, pch=4, col="red", cex=4, lwd=4)
```



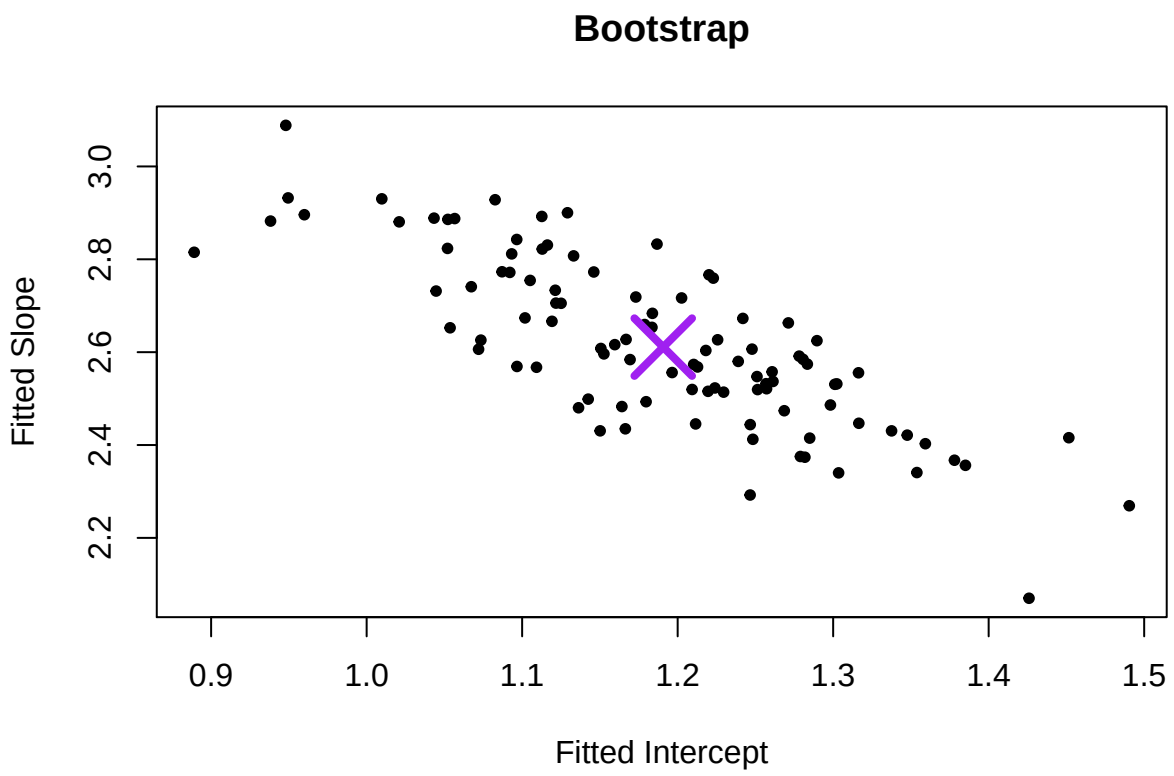
Using the bootstrap to approximate the uncertainty

In practice, we cannot sample from the true distribution because we do not know the true distribution. We only have one set of data: the 200 pairs of observations: $(X_1, Y_1), \cdots, (X_{200}, Y_{200})$.

The idea of the bootstrap is that *we sample with replacement of the original dataset to generate new dataset, and use the new dataset to obtain the estimator*. Ideally, when the sample size is large, the original sample is similar to the population (true) distribution so that sampling from the original sample would be similar as sampling from the population.

Here is an implementation for the bootstrap:

```
bootstrap_table <- matrix(NA, nrow=100, ncol=2)
for(j in 1:100){
  w <- sample(200,200, replace=T)
  X_bt <- X[w]
  Y_bt <- Y[w]
  fit_bt <- lm(Y_bt~X_bt)
  bootstrap_table[j,] <- fit_bt$coefficients
}
plot(bootstrap_table, xlab="Fitted Intercept",
     ylab="Fitted Slope", main="Bootstrap", pch=20)
points(x=fit$coefficients[1], y=fit$coefficients[2],
       pch=4, col="purple", cex=4, lwd=4)
```



The distribution of bootstrap estimators and the distribution of the original estimator look very similar to each other. This shows how we can use the bootstrap to approximate the uncertainty of our estimator.

However, the bootstrap will not always work; there are many cases where the bootstrap fails. Proving the validity of the bootstrap under various assumptions is still a big research topic in statistics.